Response Under 37 CFR §1.111

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No.	:	09/775,411	Confirmation No. 8007
Applicant	:	JARED J. JACKSON et al.	
Filed	:	02/01/2001	
TC/A.U.	:	2171	
Examiner	:	Brian D. GODDARD	
Docket No.	:	ARC920000141US	
Customer No.	:	23334	

37 C.F.R. 1.131 DECLARATION

I, the undersigned, inventors of the above-referenced patent application, hereby declare the following:

1) The pending claims of our above identified patent invention were rejected under 35 U.S.C. §102(e) / § 103 based on the prior art reference of Steele et al. (U.S. Patent Pub. No. 2003/0191737) with a filing date of December 18, 2000. (hereinafter referred to as "Steele").
2) The invention described in the above referenced patent application was reduced to a writing and signed by the undersigned applicants prior to the December 18, 2000 filing date of Steele. In particular, the relevant portion of our Invention Disclosure upon which the above referenced patent application was based is attached herewith.

We, the undersigned, declare all of the above statements are made on our own knowledge, the above statements are true and correct, and the above statements are made on information that we believe to be true. We understand that false statements or concealment in obtaining a patent will subject us to fine and/or imprisonment or both (18 U.S.C. §1001) and may jeopardize the validity of the above identified patent application or any application issuing therefrom.

[illegible]

Jared J. Jackson *	Jussi P. Myllymaki
July __, 2004	July 21, 2004

* - Unavailable for signature under MPEP § 715.04

Proxy-Based Crawler Mechanism for Forms, Scripts, and Dynamic Hyperlinks

***Idea of disclosure**

1. Describe your invention, stating the problem solved (if appropriate), and indicating the advantages of using the invention.

Web crawling is a well-established technology for creating search engine indexes and other databases. Crawlers find Web content (HTML pages and other documents) by following hyperlinks which are Uniform Resource Locators (URLs) appearing in the body of HTML pages. A limitation of today's crawlers is that they can only follow static hyperlinks, links in which the full URL is plainly visible in the HTML document and easily extracted by the crawler.

In contrast, it has been argued that the volume of content accessible via static hyperlinks is dwarfed by the volume of content accessible via HTML forms and other non-static hyperlinks (leading to the "Deep Web" concept advocated by BrightPlanet.com). It is therefore important for crawlers to be able to access this additional content, especially since the content generated in response to following dynamic hyperlinks typically originates from proprietary databases containing highly valuable competitive information. For instance, Amazon.com has a database of millions of books that it sells, yet they provide static hyperlinks (in the form of browsable categories) only to the bestsellers in different categories, not the entire database. Therefore, a crawler that only follows static hyperlinks will see only a small fraction of the entire database.

The present invention solves this problem by describing how HTML forms and JavaScript, the two primary mechanisms for dynamic hyperlinks, can be handled in today's crawlers. The result is a crawler that is able to index a vastly larger set of Web documents than before.

2. How does the invention solve the problem or achieve an advantage,(a description of "the invention", including figures inline as appropriate)?

As described earlier, Web crawlers only follow static hyperlinks, which are URLs such as " http://www.ibm.com/". An HTML page, the primary source of hyperlinks for crawlers, typically contains a set of static hyperlinks in its "<A>" (anchor) tags, "<IMG>" (image) tags, and "<FRAME>" (child frame) tags. Additionally, an HTML page can contain one or more forms that allow an end user to enter search criteria or other information, which is then transmitted to the Web server. The Web server processes the user's input, responding with another HTML page or other Web document. The user's input is transmitted to the server in one of two ways. The simplest approach is referred to as the "GET method", where a hyperlink is constructed by the user's browser -- the hyperlink combines a base URL and the user's input to form an augmented URL. The Web server receives the URL and disassembles it back into the base URL and the input given by the user. Alternatively, the Web site may request the use of the "POST method" where the hyperlink contains only the base URL, with the user's input transmitted separately.

Advanced Web sites, especially those used for electronic commerce (eCommerce) applications, require a more sophisticated control over the interaction between the user's browser and the Web server. In particular, information such as cookies, user names, session identifiers, catalog names, and shopping cart identifiers are transmitted back and forth between the browser and the server, without the user really seeing this background data traffic. The aforementioned information is typically combined with the user's own input (e.g. author name to search on at Amazon) by using client-side scripts. The foremost client-side scripting language is JavaScript. When the user submits a form to the server, a JavaScript program can, for instance, intercept the request and piggyback the request with additional background information. The Web server expects to see the background information present in the request and can refuse to provide the requested result page (e.g. list of books by an author) if the background information is missing.

While modern browsers support both HTML forms and JavaScript, no crawler that we are aware of does so. It is for this reason that crawlers are excluded from a large body of Web content that is otherwise available to interactive users via their Web browsers.

The present invention solves this problem by introducing a transcoding proxy between Web sources and the crawler. The crawler accesses the Web via the proxy, unaware of the fact that slight modifications may have been performed on the Web content by the proxy. In particular, the proxy performs the following transcoding on HTML pages: it analyzes each HTML page and detects when HTML forms and JavaScript are present in the page. For each HTML form, it computes "synthetic hyperlinks" that correspond to the possible form input values and inserts these hyperlinks as static hyperlinks into the document. The augmented document is then given to the crawler for subsequent processing.

The transcoding proxy also analyzes JavaScript code and detects hyperlinks that are a result of computation performed by the JavaScript code. Two possibilities exist for analyzing JavaScript code. One solution is for the proxy to use a JavaScript engine to actually execute the code when the HTML form would be invoked. Another, simpler but less generic, solution is to provide a tailored filter for each target Web site to be crawled. This solution is unworkable if the aim is to crawl every Web site in the world, but is a good approach for very focused crawling, e.g. indexing the contents of a competitor's Web site.

The JavaScript analysis is done in conjunction with HTML form analysis. If an HTML form is programmed to activate a JavaScript function, the transcoding proxy would also perform that action and combine the actions of the JavaScript functions with the synthetic hyperlink generated from the HTML form alone (i.e., the "user input"). If no JavaScript function is defined, the synthetic hyperlink generated from the HTML form is used without modification.

Since static hyperlinks, by their nature, can only access staticly referenced web documents and form URLs specified using the "GET method", today's crawlers are not able to access form URLs specified using the "POST method" without some form of assistance. This requires that web documents accessed using the "POST method" recieve special handling. The solution is to accompany the transcoding proxy with a conversion proxy that will take static hyperlinks augmented as in the "GET method" and convert them into "POST method" requests. The synthetic hyperlinks use as their base URL the method conversion proxy, with the ultimate target URL as a parameter appended to the base URL. The method conversion proxy dissasembles the target information (including the ultimate target URL, and "user input") and issues a "POST method" request to the target web site. The response from the web site is then returned to the crawler. Through this processs, the crawler is completely unaware of the fact that the synthetic, static hyperlink was generated from an HTML form (and potentially involved the use of JavaScript) and that a "POST method" request may have been involved. The translation process is described in more detail bellow in architecure sections 2 (Transcoding Proxy) and 5 (Method Conversion Proxy).

The architecture of the system is illustrated in Figure 1. The individual components are described below.

1. Crawler
2. Transcoding Proxy
3. Script Engine
4. Web Server
5. Method Conversion Proxy
6. Custom Script Filter

1. Crawler

The Crawler (item 1 in Figure 1) can be any one of the many currently available crawlers on the market or developed internally. Its task is to fetch one or more "seed pages" (item A in Figure 1) which are HTML pages containing static hyperlinks. The hyperlinks are extracted and inserted into a hyperlink pool. The crawler then itteratively fetches a hyperlink from the pool and fetches the corresponding document from the Web. If it is an HTML page, its hyperlinks are again extracted and inserted in the pool. The crawler is configured to crawl only to a certain "depth", in other words, it may only move a certain number of hyperlinks away from the seed pages. The distance is measured in terms of the number of hyperlinks followed, so if the depth is 3, the crawler will not go farther than 3 links away from the seed pages. Other parameters may control the speed and frequency with which Web documents are accessed, and whether every link is followed or a filtering mechanism allows for only a subset of all possible links to be pursued.

In the context of the present invention, the seed pages given to the crawler are modified so that the ultimate target Web site URLs are passed as parameters to the transcoding proxy. For instance, if the target URL is "http://www.ibm.com" and the transcoding proxy runs on the same computer as the crawler, the seed page would contain static hyperlink similar to "http://localhost/transcoder.cgi?url=http://www.ibm.com/".

2. Transcoding Proxy

The Transcoding Proxy (item 2 in Figure 1) receives a hyperlink request from the crawler and returns a Web document to it (item B in Figure 1). The proxy first disassembles the request it received and determines the ultimate hyperlink to be retrieved. In the above example, the proxy would extract the address "http://www.ibm.com/". That document is downloaded from the ultimate target Web site (item C in Figure 1) and an analysis of that document is initiated. If the document is not HTML, the analysis is terminated and the document is forwarded to the crawler immediately.

If the document is HTML, the analysis inspects each "<FORM>" element in the document and constructs a list of synthetic hyperlinks that an interactive user would be able to activate by using that form. Forms can contain three types of input elements: pulldown menus ("<SELECT>" and "<OPTION>" tags), check boxes/radio buttons ("<INPUT>" tags) and text entry boxes (also "<INPUT>" tags). The analysis combines the options listed in pulldown menus and check boxes/radio buttons in all possible ways and forms initial hyperlinks. Text entry boxes that contain default values are appended to the hyperlinks with the default value selected. Text entry boxes that do not contain a default value will be assigned a set of synthetic values. The synthetic values include single characters ("A" through "Z"), single digits ("0" through "9"), and a carefully chosen list of common nouns (e.g. "travel", "sport", etc.) These text entry values are also appended to the hyperlinks.

As discussed earlier, JavaScript code is handled either by a Script Engine that takes as input the hyperlinks and the JavaScript code of the HTML page, or by a Custom Script Filter that is customized for the Web site where the original HTML document was retrieved. In either case, the output is a modified set of synthetic hyperlinks.

The analysis of forms results in the generation of a large set of synthetic, static hyperlinks. These hyperlinks are now inserted into the original HTML page. If the method used in the form is "POST", the synthetic hyperlinks are altered so that the hyperlink becomes a parameter passed to the Method Conversion Proxy. For instance, if the synthetic hyperlink is " http://www.xyz.com/search.cgi?q=sport" and the Method Conversion Proxy runs on the same computer as the Transcoding Proxy, the augmented hyperlink would look similar to " http://localhost/conversion.cgi?url=http://www.xyz.com/search.cgi?q=sport".

Once the analysis is complete and the original HTML page has been augmented with synthetic hyperlinks, the augmented document is returned to the crawler.

3. Script Engine

The Script Engine (item 3 in Figure 1) takes a list of synthetic hyperlinks produced by the Transcoding Proxy, plus the script code (functions) defined in the HTML page. For each hyperlink, the Script Engine executes functions of the script code and produces an output that may be identical or different from the original, synthetic hyperlink, depending on the actions of the script code. The modified set of hyperlinks is returned to the Transcoding Proxy.

4. Web Server

The Web Server (item 4 in Figure 1) is a standard Web server that receives a hyperlink request, retrieves a local Web document (item D in Figure 1), and returns it (items C and E in Figure 1) in response.

5. Method Conversion Proxy

The Method Conversion Proxy (item 5 in Figure 1) receives a "GET method" hyperlink request from the Crawler, extracts the "url" parameter of that request, further extracts the CGI parameters encoded in that URL, connects to the Web Server indicated in the base part of the URL, passes the CGI parameters to the Web Server using the "POST method", and receives the Web Server's response (item E in Figure 1). The response is then sent back to the Crawler (item F in Figure 1).

6. Custom Script Filter

The Custom Script Filter (item 6 in Figure 1) is designed to handle the script code of a particular Web site or set of Web sites (for instance, those that use the same eCommerce server). The Custom Script Filter is manually programmed to analyze the script code of an HTML page and perform the actions described in the script code, without actually executing them in a Script Engine. This is preferable to executing the code in a Script Engine if a) the code is large and the execution would be slow, b) if the Script Engine is not able to execute the script code due to incompatibilities between script language versions, or c) if a Script Engine is not available.

A Custom Script Filter takes as input the set of synthetic hyperlinks generated by the Transcoding Proxy, plus the script code (functions) defined in the HTML page. For each hyperlink, a custom computation, defined by one or more filters (item G in Figure 1), is performed. The modified set of hyperlinks is returned to the Transcoding Proxy.

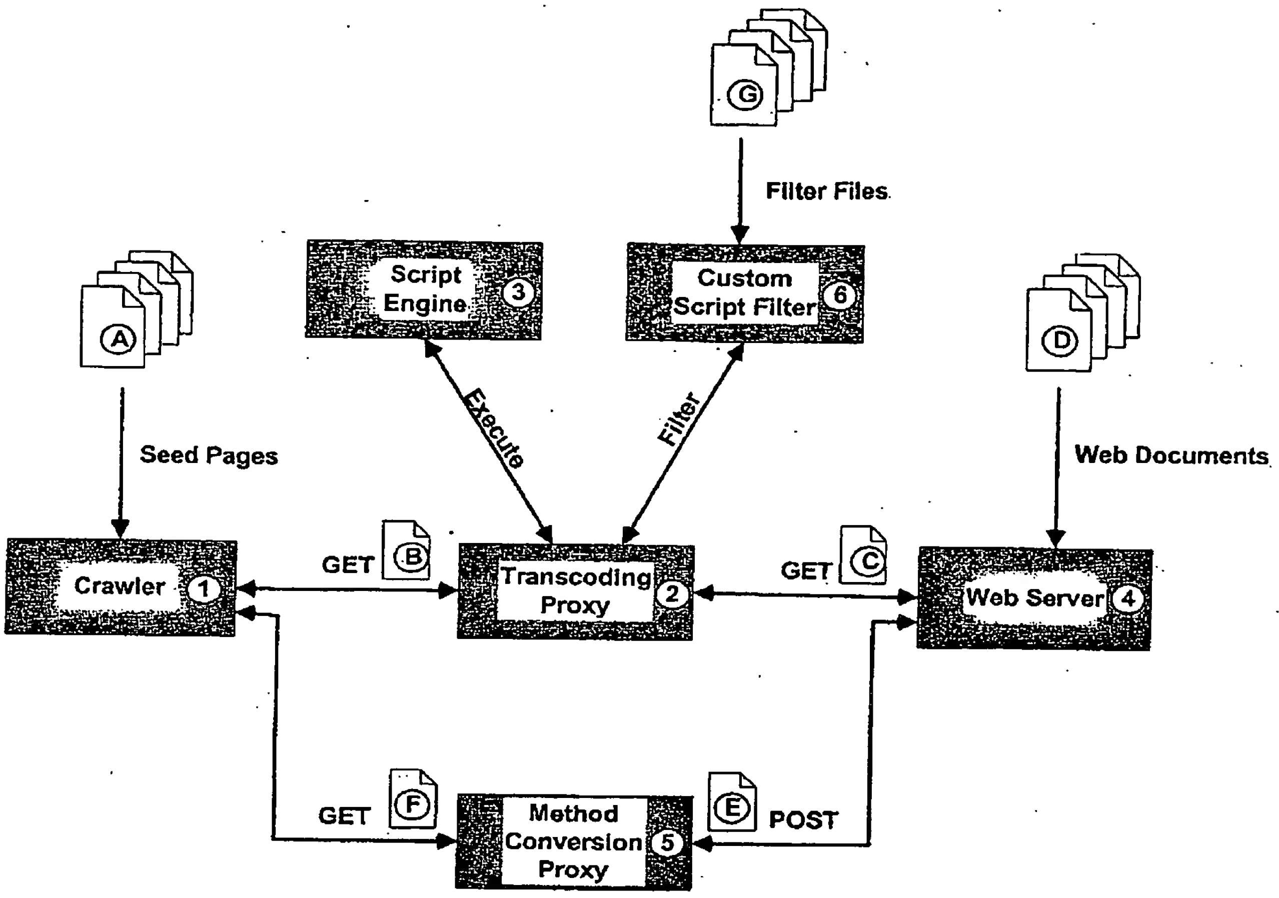

Figure 1. Architecture of the Proxy-Based Crawler Mechanism for Forms, Scripts, and Dynamic Hyperlinks.

Note: For some reason I'm not able to edit items 3 and 4 in this document. The text for those items is given below.